ESD-TR-68-143, VOL II
ESTI FILE COPY

ESD RECORD COPY

RETURN TO
SCIENTIFIC & TECHNICAL INFORMATION DIVISION
(ESTI), BUILDING 1211

ESD-TR-68-143, Vol. II

EVOLUTIONARY SYSTEM FOR DATA PROCESSING

CONTROL AND USE OF THE SYSTEM

Charles T. Meadow
Douglas W. Waugh
Gerald F. Conklin
Forrest E. Miller

ESD ACCESSION LIST
ESTI Call No. AL 61083
Copy No. / of / cys.

January 1968

COMMAND SYSTEMS DIVISION
ELECTRONIC SYSTEMS DIVISION
AIR FORCE SYSTEMS COMMAND
UNITED STATES AIR FORCE
L. G. Hanscom Field, Bedford, Massachusetts

This document has been approved for public release and sale; its distribution is unlimited.

(Prepared under Contract No. F19628-67-C-0254 by Center for Exploratory Studies, International Business Machines Corporation, Rockville, Maryland.)

AD670839

LEGAL NOTICE

When U.S. Government drawings, specifications or other data are used for any purpose other than a definitely related government procurement operation, the government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

OTHER NOTICES

Do not return this copy. Retain or destroy.

FSC 68 0669/

ESD-TR-68-143, Vol. II

EVOLUTIONARY SYSTEM FOR DATA PROCESSING
CONTROL AND USE OF THE SYSTEM

Charles T. Meadow
Douglas W. Waugh
Gerald F. Conklin
Forrest E. Miller

January 1968

COMMAND SYSTEMS DIVISION
ELECTRONIC SYSTEMS DIVISION
AIR FORCE SYSTEMS COMMAND
UNITED STATES AIR FORCE
L. G. Hanscom Field, Bedford, Massachusetts

This document has been approved for public release and sale; its distribution is unlimited.

(Prepared under Contract No. F19628-67-C-0254 by Center for Exploratory Studies, International Business Machines Corporation, Rockville, Maryland.)

FOREWORD

This report presents the results of a study of the specifications for an information system intended to support the design, production and maintenance of large computer programming systems. Called Evolutionary System for Data Processing, or ESDP, it was begun as an internal IBM project in 1965 by the Center for Exploratory Studies of the Federal Systems Division and continued under Air Force sponsorship during 1967 and early 1968.

This work has been performed under contract number F19628-67-C0254 for the Electronic Systems Division, U.S. Air Force Systems Command. The project monitor was Mr. John Goodenough, ESLFE.

The authors wish to express their appreciation for the encouragement and assistance provided by Dr. John Egan, formerly of ESD, and their colleagues Dr. Harlan D. Mills and Mr. Michael Dyer.

This report is in four volumes: Volume 1, System Description; Volume 2, Control and Use of the System; Volume 3, The CAINT Executive Language and Instruction Generator; and Volume 4, Programming Specifications. This report was submitted on January 31, 1968.

This report has been reviewed and is approved.

Sylvia R. Mayer
SYLVIA R. MAYER
Project Officer

[illegible]
WILLIAM F. HEISLER, Col, USAF
Chief, Command Systems Division

ABSTRACT

ESDP is a proposed system whose purpose is to acquire, store, retrieve, publish and disseminate all documentation, exclusive of graphics, concerned with a large computer programming activity. Documentation is deemed to consist, not only of final or formally published after-the-fact reports, but of working files, design and change notices, informal drafts, management reports--in fact, the entire recordable rationale underlying a programming system. Maximum attention has been concentrated on the means of acquiring and organizing documentation. Two major, complementary approaches are proposed. The first is called Program Analysis and is a process of <u>extracting</u> documentation directly from completed programs. The second is called Computer Assisted Interrogation and is a process of <u>eliciting</u> information directly from the programmer, through on-line communication terminals. The former provides canonical data about the program's structure. The latter provides explanatory material about all aspects of the program, and in the absence of canonical data, may provide tentative structural information as well. The conclusion of the study group is that ESDP is a feasible concept with present-day technology and that it will materially benefit using organizations in the production of programs and in guiding their evolution as requirements change. Its value will be greater for larger organizations, whose internal communications difficulties tend to cause truly gigantic inefficiencies. Its implementation as a support system for such projects would require a significant quantum of investment in order to produce these benefits and is predicated on the use of a computer system dedicated solely to the use of ESDP.

Volume 2

Control and Use of the System

I

INTRODUCTION

The concept of what is the "best" way to describe an object programming system will change and we deem it one of the major accomplishments of ESDP that the documentation system described herein can be readily adapted to suit these changing requirements. In use, ESDP would be adapted to the particular requirement of each using organization or project. We do not feel there exists a fixed, standard way to document a computer program or programming activity. On the contrary, the documentation requirements depend upon the people involved, the purpose of the system, and requirements imposed by law, contract or administrative regulation, all of which are in constant flux.

At present, there is no measure of effectiveness of program documentation. The ability to change documentation procedures in response to changing requirements, while intuitively desirable, does not of itself guarantee effective results. For example, there are any number of flow charting techniques and automatic and semi-automatic programs for producing them. Yet, a flow chart is a useful means of communication only if it can be read by a user, if it describes what he needs to know, and if it is a true representation of the current state of the program it describes. This is equally true of the various text documentation systems available. The existence of an automatic editing program does not guarantee high quality text. Some systems rely upon comments embedded in coding, but comments tend not to be sufficiently general and are rarely updated by a programmer. It follows that automatic aids to documentation do not , of themselves, accomplish the job. There remain the needs for skill in using the aids, in expressing one's ideas, in providing leadership, evaluation, and controls to the people who will perform the documentation tasks. There is also a need for the project leadership to recognize what information is needed (and, almost as important, what is not needed) to satisfy members of the project, its sponsoring organization, and the ultimate users of the programs.

Automatic documentation does help to achieve document standardization. ESDP offers a variety of automatic documentation aids. It offers positive assistance by performing some of the documentation, through program analysis, and by guiding the programmer in his narrative documentation through computer assisted interrogation. Both these forms of information acquisition, as well as the associated retrieval, and dissemination programs, can be modified by project management to suit their needs. Still, we repeat, _the skill of the user and the leadership of his management are essential to effective documentation_.

Control over ESDP is exercised by modifying, or adapting through programming, a number of system elements to change the system logic, or by administrative instructions imposed upon users by management.

1. Programmed Controls. The principal programmed control points are:

a. Control of documentation files - the catalog of information acquired and maintained.

b. The source dependency of documentation - basically, the matter of which people or programs contribute to the documentation of an object program or data element.

c. Storage of program text - the actual source language coding and modifications to it.

d. File management - control of access to files, authorization for change, reconciliation among information items from different sources.

e. Output produced - scheduled, standard reports and ad hoc reports.

f. COMPOOL management

g. Dissemination control.

2. Administrative Controls. The principal points governed by administrative regulations are:

a. Determination of what information is needed and what reports are required.

b. Frequency and conditions of documentation and updating of documentation.

c. Freedom of use of report generation and information retrieval facilities.

As a documentation, hence communication, system ESDP can also be a management tool. That is, it is useful not only for describing programs but as a vehicle for exercising control over their design, production and modification. The manner in which the facilities are used is, of course, another important variable in the application of ESDP to any particular project.

II

SYSTEM CONTROL

1. Executive Programming. Most of the programmed controls are exercised through the medium of executive programming in the CAINT Executive Language (CEL). Through this language (See Volume 3, Section I) systems managers can control the acquisition of information through interrogation, dissemination, document production and instruction. At the present state of development, the features of ESDP not under executive program control are program analysis, some aspects of graphic composition and some aspects of text editing. The last two services of ESDP would, we expect, be implemented by using existing programs, such as FLOWCHART [1] or Administrative Terminal System [2]. Users would communicate with these programs, i.e., specify what these programs are to do, through an interrogation which is conducted by an executive program. Both these systems accept user macro commands. These commands would be acquired by ESDP through an interrogation. The ESDP user or manager could modify the manner of eliciting macros or establish and enforce conventions for their use, but could not, without extensive reprogramming, change the programs which execute the macros.

Program analysis is the largest single program within ESDP and will probably always dominate the system in sheer size. We foresee the day when program analysis logic may also be specified or changed through an executive program, but a requirement for this capability will not be specified at this time.

The authors, or at least the designers, of executive programs will be systems management personnel or senior system analysts. They will not always be professional computer programmers. Hence, we must find a balance between the desire for a language simple to use by nonprogrammers yet expressive, or powerful, enough to do the job required. CEL, as a subset of PL/I, partially meets this objective. The language is easier to learn and use than full PL/I; however, it still requires some computer programming ability. An ideal solution might be to further simplify its use.

But, rather than alter the programming language, we can facilitate its use by providing computer assistance to executive program authors, essentially computer assisted programming. A conversational program, or interrogation, to provide this assistance would, itself, be written in the executive programming language. This would be an extension to the current ESDP concept, but could be implemented without any fundamental modification of the system programs. The program that elicits an executive program would converse with the author about results desired, options open at any time, required syntactic elements, etc. While it would not make a qualified programmer out of a sow's ear, it would reduce the number of errors made on a purely procedural level and, of course, reduce the number of unimportant rules the programmer must learn.

The Instruction Generator, described in Volume 3 and intended primarily for generation of instructional programs, can partially accomplish this objective. An extension of that program could result in an effective interrogation program generator. Opportunity will be provided for individual ESDP users to request ad hoc, or individually tailored, reports for their own use. Changing a system-wide information acquisition or reporting program to accomplish this would be intolerable.

2. Content of Documentation Files. Determining the content of documentation files is essentially equivalent to deciding what information is needed to document a program system. As we have stated, it is our thesis that there is no fixed answer or method, that each organization using ESDP will want to individualize its documentation somewhat, just as it does without ESDP.

The structure of the documentation files used to date, and planned for the future is, we think, based on the fundamental nature of programs and will not change much. The individual data items, the degree of detail, the form of representation, all are subject to much variation. For example, the question of how much narrative description should accompany each use of each data item in a program is open to debate, while the existence of a narrative description of the data item in general is mandatory. Here is where executive programming can be used to vary data acquisition. When an interrogation program recognizes that the existing documentation file contains information that item X is used in an IF statement to decide on what branch to take, the interrogation may follow a number of paths for X, such as:

- Verifying that the currently stored description of X fits the usage here, in the programmer's opinion.

- Asking for a physical interpretation of the values of X that lead to the true and false paths.

- Asking whether either condition represents an error condition.

The executive programmer can easily patch these questions into an interrogation program that originally did not have them. Similarly, if present, they could be deleted or their form changed. Such changes call for a modification to the structure of the documentation files, a slightly more complex problem, but one that can be handled with a single transaction invoking the information retrieval system, if a file directory is used. The latter program must establish a new file with a physically different structure; then, record-by-record, equate the old and the new files.

A change in documentation files will probably necessitate a corresponding change in output, or report generation, programs as well. These changes are also made by modifying an input table to an executive program.

a. Source Dependency of Documentation

Any given unit of programming has an interface with subordinate, superordinate, and parallel programs and with data structures. Thus, in one way or another, far more than one person may have a hand in specifying what a program is to do. There are two primary information acquisition techniques - program analysis and interrogation - for use with each object program. Personnel changes further add to the list of sources that may contribute to the documentation of any unit of programming or data.

Experience dictates that no one source - no one person or program should be assumed to be more reliable than another. Errors can be made in interrogation responses as well as in program code. The detection and correction of these errors in documentation files may be a subtle and difficult process. It is a process that should be performed by a qualified person but, at best, it is always subject to some chance of error.

Our recommendation is to enable each information source to create the records it needs, with conflicts resolved through reconciliation, not through automatic precedence of one source over another. The method of resolving conflicts is something which each using organization may wish to decide for itself. Precedence can be given, through executive programming, to any source or combination desired by system management. One might, for example, automatically resolve any conflicts in favor of program analysis data, on the grounds that it represents what the programmer actually <u>did</u> as opposed to what he <u>thought</u> he did. On the other hand, the automatic detection of conflicts is no small matter, and two subordinate UOP's that appear the same to program analysis might be different in fact. For example, in:

```
IF (A > B) THEN DO; A = A - B; GO TO LABEL; END;

     ELSE DO; A = B - A; GO TO LABEL; END;
```

The 'THEN' UOP is indistinguishable from the 'ELSE' UOP by program analysis techniques. A programmer could easily get the two forms confused and design his program one way but execute it another. In the final analysis, only an informed, skilled programmer can review all versions of documentation and select the data to be used.

b. Storage of Program Text

The major alternative here is whether or not to store source code within the ESDP system files. Doing so can provide a great service to users but imposes costs and restrictions both on them and on overall system operation. If ESDP is to perform incremental program analysis, then programmers must be able to submit changes to existing programs for analysis. Unless this is done by use of ADD, DELETE, and REPLACE commands, _at the statement level_, a great clerical burden is place on the programmer. If changes are submitted this way, then ESDP must maintain the program text and all changes, and must be able to find the lowest level UOP not affected by a given change and resubmit this to program analysis. As a manual process, this would be fraught with errors.

We make the assumption, then, that if program text is to be maintained within the system, changes are to be submitted as file changes. If program text is not maintained within the system, it would probably be best not to perform program analysis until after the program has been completely debugged. Returning to the first assumption, that incremental programming is to be controlled by ESDP, we must recall that any program might have more than one version in use or in debugging at any one time. ESDP, or any program text storage system, must recognize this as a fact of life and cope with it effectively. One approach is to store the original program, then keep a permanent file of changes to it. New changes would be made giving any previous change level as a reference, and any version of the program could be retrieved on demand.

Maintenance of object, or compiled programs is a separate consideration. The computer operating ESDP is not necessarily (in fact, almost certainly not) the computer in which the object programs will operate. Hence, the ESDP computer cannot necessarily compile programs for the operational computer, and, since changes are made to the source deck only (we are still assuming PL/I as the programming language) there is less need for ESDP to maintain object decks.

Each version of a program, in addition to being tagged by the program name and modification date, should have additional retrieval descriptors: name of programmer (who may not be the same for each version), date, use or purpose of the version (e.g., there might be a separate version of a program created solely to help debug another program), data base to use, etc.

c. File Management

Certain file handling tasks are system management functions. These include contributing to or limiting access to files, making provisions for multiple sources, and making copies of files for insurance or for use in program debugging. Mostly, these are items which can be changed by use of an executive program and are represented as items in the system file directory. Access can be controlled by keeping a list of authorized users and refusing retrieval service to anyone not on the list.

Each file maintained by ESDP may have associated with it a list of authorized users and another list of people authorized to change the first one. This can apply to documentation files, graphic files and program text.

Each documentation file may have any number of authorized versions of a logical record, one for each authorized source. Different rules of precedence and different numbers of sources may be used for each type of record.

3. Report Content. It seems reasonable to assume that any ESDP using organization will, at the start of a project, establish the format and content of all reports that are going to be produced. It also seems reasonable to assume that the initial plans will change somewhat as experience is gained on the particular project. Systems management can control or vary the nature of the outputs produced in the following ways.

a. Specification of standard system documentation, such as program design specifications, operational program descriptions, test plans, data file descriptions, etc. Specifying a report requires listing the specific data items to be contained within it, the structure of the document, its format, and its frequency or conditions of issue. Control and structure information is stated in terms of information element numbers (IEN's) which identify each discrete item of information in the ESDP data base. This information is provided as parameters to a report generating program written in the CAINT Executive Language.

Each element of the ESDP data base is assigned an information element number. An IEN consists of the IEN of the higher order information element (if such exists) and an index number among all elements at the same hierarchic level. A report form, whether standard or ad hoc, is composed by supplying to ESDP the report structure, in terms of IEN's, renumbered for use on the report, and the relationships of the report IEN to the data base IEN's. For example, a report composer may wish to start his request with data base IEN 2.3 but he would like this information to be tagged 1.0 in his report. He need only supply the report generator these two numbers.

Most likely, users will design several standard report forms and generate these periodically. Examples of standard reports might be:

Program description for each 'independent' program.

Program module description--for groups of related programs.

Data base description--for all data sets used in the object system.

Error procedures--the error procedures extracted from each individual program and cumulated.

b. Requirements for special or ad hoc reports can be specified and generated in the same manner as standard reports. It might occasionally be useful to compile a report covering all programs using a certain data file, or all concerned with performance of some function discussed in nonprogramming terms, such as federal income tax computation.

c. Standard or special report specifications can be changed to vary the content or structure of a report. This means elements can be deleted from reports or changed in position within the report. Of course, the capability is here to split information out of a report structure altogether and create a new document or document series for it.

d. Adding new information to a report is possible but presents two distinctly different degrees of difficulty, depending on the composition of the data base. If the new information exists in the data base, but has not previously been used in connection with report type t, then t can be readily modified to encompass the data. If, however, the data is not already in the data base, a significantly different problem is presented. To add a new item of information not in the data base, the data base structure must be changed and an interrogation program must be written or an existing one modified to elicit the new information. If the new item in any way affects the elicitation or interpretation of an existing item, more than one interrogation may have to be changed. In making any such change, computer assistance can be provided to help determine which interrogations acquire or refer to a data item, or in which reports an item is found.

e. Certain information cannot be entered into ESDP. An example is a sketch needed to illustrate program logic. Since ESDP as presently conceived has no graphic input capability this form of input would best be handled by establishing a separate IEN for each non-storable item and including these in a reference citation to the item that tells, also, what it is and where it is. Flow charts and tables can be handled by ESDP, by providing information from the ESDP data base to a standard flow chart or table generating program.

4. Dissemination and Communication. Dissemination in the context used here refers to the activity of transmitting or delivering information to interested persons without their having to make an explicit request for each item of information. Although dissemination can be handled on a request basis, such requests would be standing requests, asking that all information that becomes available to the disseminating agency be made available to the requestor. We feel that a dissemination system for a large programming project is a necessity to ensure that, when changes are made to programs, files, specifications or

schedules, all those concerned are promptly notified. On the other hand, we recognize that dissemination of too much information may defeat its own purpose. If each recipient gets so much material in a day that he no longer has time to read it all and do his own job, he will probably stop reading or read only selected (possibly randomly selected) material. Thus, much of the effort going into dissemination would be wasted. To assure adequate dissemination but guard against flooding, controls must be provided to enable systems management and individual recipients to monitor and control the flow of information.

a. Dissemination Techniques

The most common technique used in dissemination is to establish a file of user, or recipient, profiles and to match these profiles against new information, transmitting the new documents or notice of their existence when a match is achieved. The profiles are usually stated in terms of key words or classification terms, or, in the case at hand, program or file names, and the documents are indexed in the same terms. Documents in ESDP may be individual information elements or any grouping of them. There are a number of variations of this basic technique:

(1) Each member of a programming project defines his own interest profile by listing the names of programs or files in which he has an interest, or key words defining subject areas of interest. In the simplest form, it would be incumbent upon each user to assure that he lists all interfacing programs and files. Mechanically, this is a simple system to implement, but it has the weakness that the individual must anticipate what programs and files he will become interested in and if he does not continually update his profile as system design and implementation progress, his output is of progressively less value.

(2) A variant on the previous method that is self-adaptive to changes within the object program system would be to expand the logic of the profile records to permit a user to specify interface information parametrically. He might name a program of interest (say his own) and ask for information pertinent to it, any program that interfaces with it, any program that produces files used by that program, etc. The determination of which are the interfacing programs or files can be easily made through the system data base, and, as changes are made in program and file structure, the effect of a profile may change.

(3) Systems management might choose to modify the previous procedure by requiring that the name of each program a person is assigned to is automatically placed in his profile, together with functional modifiers that specify data files used and interfacing programs. Management may, at its option, assign, permit or deny access to information concerned with performance of individuals, project cost data, and other sensitive items.

(4) The creator or modifier of any information item might wish to specify certain recipients, regardless of whether or not they have requested this information through their profiles. This specification can either be by name or functionally, by denoting program or file names and modifiers to those names.

(5) Finally, all the methods and variations would permit dissemination profiles to be changed readily, as interests, assignments, status of the object system, and performance of the dissemination system vary.

b. COMPOOL Management

In large, highly interconnected programming systems, centralized control over data specifications is a necessity. It is not our purpose to justify this approach to project management but to illustrate some of the documentation and communication problems that arise in connection with the object system's data base, and techniques available to overcome them.

Two programmers planning to use the same file may, early in the design stage of a project, assume different details of content or structure to suit their individual preferences. Or, they may each define a different file with essentially the same information content, each being unaware of the other's work. One programmer may find it necessary to change a file structure and then it is imperative that those changes be made known to other users of the file and conflicts resolved. Certainly, not all proposed changes can always be permitted, for their consequences to other programs may be intolerable. More subtle communication problems arise in making sure that all programmers concerned use the same value of constants and observe the same validity restrictions on data values.

To handle these communication problems, ESDP can be used by COMPOOL managers to acquire information on data requirements and plans, to detect conflicts, duplication, and omissions, and to communicate to the programmers concerned. The plan, then, is for each programmer to record his data usages as a routine part of program documentation, in all phases of system development. This information will be collected and scanned by the COMPOOL control group who will insure adequate documentation and resolve conflicts. Finally, decisions will be promulgated to all concerned, an action which may trigger another round of analysis. Thus, acquisition, analysis, synthesis, and dissemination are continuously in progress, particularly while the system design is being evolved. Once the specifications become firm, changes will be less frequent, but more critical when they do occur.

We have stressed elsewhere how each programmer defines his own data and how ESDP may acquire multiple records about each data item or file. Ideally, each user of a file or item would define and structure the data the same way, but, in fact, this will not happen. The COMPOOL controllers can use the reconciliation to review the different versions of data specifications, make and disseminate decisions. If they choose to defer judgment until later, all information, from all sources, can be retained until a decision is made. Immediate dissemination of COMPOOL reconciliation decisions is vital, for even small imposed changes to data specifications can result in program changes out of all proportion.

This process is definitely iterative. Not all aspects of the problem will be mechanized. Much of the work of resolving differences will be carried out in personal discussions. What is important is that as soon as any programmer documents any part of his data the implication to all other programmers be immediately made known, and once the COMPOOL controllers have made a decision, that decision be promulgated to all interested parties. Thus, the acquisition of information is decentralized--it is a function largely of the individual programmers--but its control is centralized.

We must point out that control over the object system data base is limited to control over specifications and documentation. Unless ESDP and the object system are integrated and run in the same computer (or connected computers) ESDP cannot control the content of the object system data base.

By similar methods, COMPOOL and other systems managers can monitor and control such data-related areas as auxiliary storage allocation, I/O and storage device utilization, and core memory utilization.

5. Management Controls. Many of the controls on the use of ESDP must be exercised by management through administrative regulation. The most important of these controls has also been discussed under programmed control-determination of file content and subject of interrogations. The decision on what information to collect is obviously made by people, not a program. The implementation of these decisions is done through programs and file organization, hence was discussed in the preceding section. What is important to realize here is that these decisions are not permanent, they can vary from project to project and, to some extent, within a project. Particularly during the design phase of a project, management may feel it necessary to vary the information being collected or the manner of collecting it. These changes will be easy to effect, but it must always be borne in mind that a program change involving a file organization change may be a large undertaking. Interrogation changes, or simple additions to an existing file are less difficult and may be made with less chance of system disruption.

The remaining administrative controls center around frequency of use of the system and freedom of access to information.

a. Frequency and Conditions of Use of ESDP

A major management decision is whether or not to use design interrogation, and if it is used, to what extent. Secondarily, there are considerations of how often to update documentation, whether it be during design or after completion of a program.

If design interrogation is used, management must decide how often each programmer or system designer should update his current documentation. It is our feeling that this should be done often, as the primary purpose of design documentation is the rapid dissemination of change information, in time to help other programmers adapt to it. Once a week, at least, and perhaps once a day appear reasonable choices.

Ideally, a program should be run through program analysis only after it has been fully debugged. Otherwise, interrogations are generated on the basis of possibly erroneous information and a quite complicated error pattern can be built up in the documentation files. On the other hand, it may be necessary to have substantially complete documentation before the program is that close to completion. The earliest feasible point at which to run program analysis is just after the first successful compilation. Program analysis (PA) should never be run on a deck that cannot compile because the PA program does not contain as many error checks as the compiler and syntactic errors in the program may confound PA. If the earliest opportunity is taken for PA, then the changes during debugging must be reanalyzed and redocumented. It would then, be a management decision how often to update the program analysis files and reinterrogate. One possibility is to make such an update each time a change is made to a program deck, or each time a new compilation is made. This practice would result in an up to date documentation, but might be quite time consuming and expensive. Other possibilities are to schedule a reanalysis and reinterrogation periodically, such as every month, or require just one analysis at first compilation and one more at the completion of the program. Only experience and the particular needs of the individual project can dictate these choices.

The retrieval, report generating and dissemination facilities of ESDP make immediate review of documentation by appropriate supervisory personnel possible. Upon review of a document or portion thereof, a manager can comment on any item contained in it, suggest changes or require changes. By simply changing the value of what could be a one-digit tag, the reviewer can indicate that he is attaching a comment for consideration by the originating programmer, is suggesting a change which the originator _may_ wish to incorporate, or is _requiring_ that a change be made. This tag can be scanned by the interrogation program next time the originator is being interrogated and used as the basis for forcing a review by the originator of the comments that management has made on his previous work.

b. Access to Files

In general, unless security regulations indicate to the contrary, it seems desirable to have all technical information on a project available to all personnel participating in it. Aside from security controls which are not considered here, two forms of access control may have to be imposed. The first is control over changes to documentation files; the second, access to financial or personal information (such as performance figures) not intended for general dissemination.

By now it is almost standard practice to have access lists containing the names or serial or identification numbers of personnel who are permitted to retrieve information from certain files. While lacking the rigor of a military security control, this is an inexpensive and reasonably effective method of accomplishing the objective. The same basic technique can be used for file maintenance, to assure that changes are made to a file or to a record of that file only by an authorized person. For example, it might be decided that only the programmer assigned to a program is authorized to change the documentation for that program, while everyone is authorized to retrieve technical information about it, and only a small list of management people are allowed to retrieve information relative to previous predicted and actual milestone achievements.

The same situation exists at the level of file organization and report generation. While the documentation files in ESDP are intended to be changeable and a flexible report generation system is to be provided to permit recovery of the report needed at any time, certain system standards will be set up, for any project, and great care must be taken that these are not changed, by malice or accident. An inexperienced person making a change in file organization or in the composition of a system-wide standard report generator or interrogation may inadvertently remove a field from the file, or fail to consider all aspects of acquisition of the information. The result may be actual loss of existing or expected information, or acquisition of logically incomplete information.

To cope with the problem of file organization, interrogation program writing, and report generation, a privileged mode of operation of ESDP can be used. Use of this mode would be restricted to authorized persons in the same manner that access to files can be restricted.

III

USER'S GUIDE

1. Steps in Documentation. Programmers, systems analysts and managers should periodically review the current documentation on their programs, projects or files and make whatever changes are appropriate. The system has been designed around the assumption that rapid dissemination of design changes are invaluable, both to the using organization as a whole and to each individual. This can only be true if updating of documentation is done conscientiously and is mechanically easy.

Because the system is organized around incremental documentation, the minimum demand is made upon a documentator's time and patience. Still, we must expect, hence design for, errors, conflicts, and redundancies.

Redundancy is not an undesirable attribute of good writing. On the contrary, if a report is restricted to presenting each item of information once and only once, it would be difficult to write and to read. What we must do in ESDP is recognize which information elements are liable to be redundant with which others so that, when a redundant element is changed, each repetition is changed also. For example, there are several different places where the purpose of of a UOP is described in an ESDP program report. It may be as a subordinate of a higher level UOP, in the introduction to its own report, and, in aggregation, in its subordinate UOP descriptions. It is only important that the designers of the interrogation program understand this and see to it that all versions are updated when a change is made in any one of them.

Through automatic dissemination or through retrieval initiated by the documentor, each programmer can find all references to interfaces or common usages of data in the documentation of other programs. These should be reviewed carefully before or during updating sessions. It is not possible to write a program that will always be able to detect conflicts in documentation. Ultimately, it will be the individual programmer or analyst who must detect conflicts and take action to resolve them.

The most common mode of usage of ESDP, then, will become that of the individual programmer reviewing his previous documentation, reviewing related material written by others and making the changes required. It should be possible to accomplish this in as little as an hour or two a week, certainly not a major demand upon a programmer's time.

During the earliest phase of the documentation of the system, or of any individual program or file, very little information is available for any UOP or UOD. We feel it is important to document what is available, regardless, since even preliminary ideas may be of value to others. Even if a programmer can say no more about his program than that it consists of three parts: input, compute, output and uses data sets A and B to produce set C, he gives information of value to anyone who must interface with his program or in any way make use of data sets A, B and C. Even at this level, each programmer can begin to make completion date estimates, describe his general approach to the problem and provide background material for others who are planning overall data base design, memory allocation or running time specifications.

As more information is added to program structure, related information on data used and memory required is built up. Data documentation is particularly important. A program cannot be completely defined unless its data is completely defined. Data structures are often the primary means of communication between programmers and the external world of those who specify what a program system is to do. Data is also the medium of interface between programs.

Data documentation requires far more than the data specifications contained in a high-order language program. It is not sufficient, for example, to know that a variable is alphabetic (or a character string). It may be necessary to know all possible values of the variable, particularly if it is a coded item, and it may be equally important to specify what action is to be taken in the event an illegal value occurs.

Since data is the primary means of interface between programs, the data structure the programmer plans on may not be the one he eventually uses. Compromises must be made with other programmers, with memory availability, with running time, with accuracy specifications, etc. Regretfully, it is not unusual for a programmer to have to make an extensive revision of a program because of an imposed change in data structure which, while not affecting the overall logic of his program, is incompatible with his original implementation. Changing the size of a record, the sequence of a file, or the manner of representation of a variable can cause such disruptions.

Gradually, as the programmer improves on and revises successive editions of his documentation he should evolve a program, or file description as complete and detailed as that obtainable through program analysis.

The difference, of course, is that design documentation records the programmer's _intent_, while program analysis records his _deeds_. It is almost inconceivable that a program of any complexity will be written and documented without some conflict or inconsistency between design documentation and program analysis documentation. This may be as small as a spelling error which causes a difference between two references to the same program, or it can be a major program reorganization showing up in program analysis but never documented through design interrogation. Therefore, there is the need for reconciliation--

combining and reconciling differences between the two forms of documentation of the same program. Reconciliation is time consuming because of the study time involved in comparing the two forms of documentation, not because of the mechanics of performing it. For this reason, it would appear to be desirable to perform reconciliation only once per program and that implies that program analysis not be performed until after the program has been debugged. However, program analysis data will be quite useful in debugging and test planning. We find, then, that the results of program analysis are useful early in the development cycle but the cost is high. Once again, the decision on when and how often to schedule program analysis should be left to the management of individual using organizations.

In planning a test of a program, the programmer can make use of the detailed documentation he has produced all along. We described how this process might work in Volume 1, Section IV.2. For now, we make the point that this is a potential major use of ESDP as an analytic tool. This is particularly so for those projects for which formal test plans and test documentation are required. The tendency to require this is increasing, particularly in large, command/control systems which it has been the objective of ESDP to serve.

2. Document Content. The steps in establishing file and document content, assuming a start from scratch, are these:

a. Design the "standard" reports to be produced for the project. Generally, this will mean all end-item reports, design notes or other publications that will probably be issued repetitively during the development of the project.

b. Combine all items of information into an ESDP data base specification. Identify which items are to be acquired by interrogation only, by program analysis only, or both.

c. Plan and write interrogation programs to elicit the information to be gotten from interrogation. Include updating interrogation which, in general, will ask subsets of the same questions as initial interrogation, but may have special review questions.

d. Plan and write, as a CAINT Executive Program, a reconciliation program for combining interrogation and program analysis results.

e. For each scheduled report, develop a table of equivalencies between report element numbers and data base IEN's. This is all that would be necessary to generate a report-writing program which would be pre-written and only in need of this table as an input.

In order to modify a document format or file structure, each of these steps must be repeated only as much as necessary to effect the change. A careful review would be needed to insure that information to be used in a report is present in the files and acquired by one or both the information acquisition systems. It is particularly important to correlate the type of data structure within the ESDP data base--to insure that an array is set up when a series of items is to be acquired, or that a structure have a separate IEN from its constituent parts.

Appendix A shows the report outline used for program logic reporting in our experimental work. In this case, because of limited memory, report element numbers are equivalent to information element numbers.

Appendix B is a summary of questions used in an abbreviated interrogation program to acquire this information.

Appendix C shows the beginning of a table of interrogation questions to be asked when updating the IEN given as argument. Also shown are question numbers, keyed to an experimental interrogation program, for eliciting information.

Appendix D is an excerpt from the interrogation on a particular program. For the purpose of clarity, we have retyped the text that actually was transmitted through a cathode ray tube terminal.

Finally, Appendix E shows an abbreviated sample of actual documentation produced through the early-model interrogation program. Appendices D and E appeared earlier as illustrations in Volume 1.

3. Updating. We have stated in Volume 1, Section II, that the program logic of file updating through interrogation is to execute only those questions concerned with the element being changed. For this reason, all the coding concerned with a question is treated as a subroutine, and subroutines can be executed in any sequence. Let us now consider how the programmer handles updating.

The first step is to decide what information has changed or what recorded items must be changed. (There is a difference--documentation may change for several reasons, a change in program logic being only one of them.) However, the same result can be achieved if the programmer will ask for a standard report to be generated on his material and do his review from that document. If he then has questions about related programs, he can use the terminal for retrieval of his answers. Whichever method is used, the programmer should come to his updating session prepared with knowledge of what items he wants to change and what new information he wants to enter.

A technique used experimentally, which we recommended using operationally, is to tie certain information elements together so that a programmer cannot change one without having to review and re-answer the closely related items. For example, if a change is made in data type (fixed, character, etc.) then such items as precision, base, etc., must be redocumented and a careless programmer should not be allowed to change floating to fixed without adjusting all the related descriptors. A response of NO CHANGE might be allowed to reduce the tedium when changes are actually trivial. In implementation, when the programmer indicates he wishes to change item data type he will actually be assumed to have asked to change the higher order structure description of a particular data item.

In our work to date we have always required a change or review of all information elements whose IEN's were equal or subordinate to that which the programmer designated as the item to be changed. This requires that a structure have an IEN and that each subordinate element within the structure have a distinct IEN.

Performing a reconciliation between interrogation- and program analysis-derived data is quite similar to ordinary updating from the viewpoint of the user. Once again, he should review his material and know what he wants to do before starting.

The object of reconciliation is for the programmer to review two, probably different, descriptions of his program and combine them into one version. If this were a simple matter of spelling errors, reconciliation would pose little challenge. However, there may be major differences in not only name but structure and it is the task of the user to bring the two together. To assist him, he will have use of a command that will enable him to lift information elements, singly or in a set, from one file and place them, differently numbered, in another. The programmer may have designed the program shown in Figure 1, but written it as shown in Figure 2.

Only the description of program D is left unaffected by what could be a change of one statement in a program which results in a change in the structural relationships among A, B, B' and C. Item-by-item, the programmer must rearrange his design documentation to conform to this new structure.

It is important to recognize that, although facilities are being provided for detecting and correcting the differences between intent (design) and fact (the completed program) it is not the intent of ESDP to provide an automatic quality control system. In other words, we cannot write a program that will be able to tell which version is correct if the two disagree and it would even be difficult to detect differences automatically beyond the obvious ones that the programmer can also spot easily.

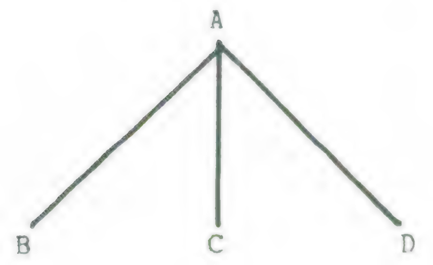

Figure 1. A Schematic Program Diagram

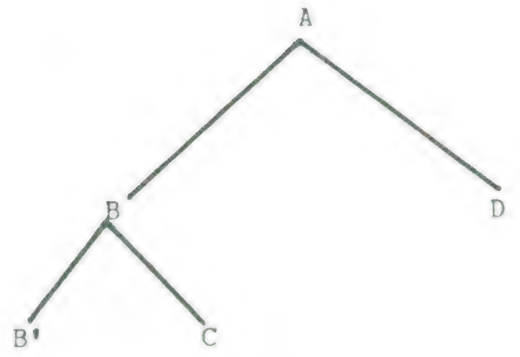

Figure 2. A Modified Program Schematic Diagram

4. ESDP Reconciliation Method.

In ESDP we propose a man-machine approach to reconciliation. Reconciliation is accomplished by scanning the various forms of the documentation of a program or data element, and using a set of commands to reorganize or replace the information. The system user may prepare for a reconciliation session by getting printed reports on each different version of his documentation. His commands then are entered in terms of the report IEN's found in these reports. He may also, or instead, use his on-line retrieval capability to stream the desired information past himself on a CRT, halting when necessary to enter a command. These commands are as follows:

(1) REPLACE ___ with ____

This command effectively moves text from one structure of the ESDP data base to another. The blanks in the command will be filled by IEN's. Since IEN's are hierarchical, one REPLACE command may be issued that will link entire substructures.

Another form of this command will allow for insertion of ranges rather than single IEN's. In other words, one could specify 4.2.1 - 4.2.6 to mean that all IEN's between 4.2.1 and 4.2.6 would be operated on, but not 4.2.7.

(2) REINTERROGATE ____

This command causes the file entries to be tagged for reinterrogation. It should be used when reconciliation cannot be performed using a REPLACE command. This means that none of the text currently in the files is appropriate and that the programmer must be interrogated again to gather the text. Again the blank is to be filled by an IEN or a range of IEN's, and the same treatment of the hierarchy is employed.

(3) TEXT ____

The TEXT command allows the user to supply appropriate text immediately rather than waiting for a reinterrogation. The same type of IEN designation will be used. Essentially, this is a command available to a very experienced system operator to allow him to provide answers to questions, without waiting for the questions to be asked.

(4) AT_____ SEE ALSO_____

This command creates a reference within the files. It is used when it is desirable to keep more than one version of documentation and provides a link between versions. The reference is stored with the first IEN given and serves as a permanent link to the other.

Since the program analysis-derived files should match the current version of the program, ESDP should not allow changes to them during reconciliation, other than the insertion of tags generated by AT______SEE ALSO_____ commands.

A feature to be included in the interrogation portion of ESDP that will aid the reconciliation process is the ability to answer any question displayed during interrogation with an IEN. This will be interpreted by the system to mean the answer to this question is stored in the system under the indicated IEN.

An example of the use of these commands in reconciliation is shown in Figure 3. The column headed REPORT (DESIGN INTERROGATION) represents a portion of a printed report from ESDP. The information in the report was derived entirely from responses to a design interrogation. Each item of information in the report is labeled with a report IEN and a title. The report is in ascending order of IEN. This report would exist in hard copy form and would be referred to by the user during the reconciliation process.

The DISPLAY column represents information displayed at the user's console during reconciliation. This is not necessarily the format to be used but indicates the kinds of data that will be of interest during such an operation. When the subsystem labeled CALC has been processed the display will be changed to indicate similar data about the next UOP.

The ENTRY column indicates the commands as they would be entered by the user at the console. The numbers to the right of each command are merely references to footnotes that describe the net result of each command.

5. <u>Assistance in Documentation</u>. To a user of ESDP documentation, its big advantage is that it provides complete, up to date reports. To a person doing documentation (who is, of course, also a user) the big advantage is that ESDP simplifies the documentation process in the following ways:

It organizes information for the documentor so he does not have to create his own outlines.

It cross references his material for him so he always has an index to his reports and that index is always automatically updated as the text is.

It provides immediate access to relevant information in other people's reports.

It enables him to make changes with a minimum of effort and reproduces a modified report almost immediately.

Some other aids to speed and ease of publication are:

```
REPORT (DESIGN INTERROGATION)        DISPLAY                           ENTRY
                 REPORT NO. 75                                      USING REPORT 75
                                    (a) CALC                          REPLACE IEN 2.3.1-2.3.1.3.2
4.2.2 SUBSYSTEM NUMBER                                                 WITH 4.2.2--4.2.2.3.2      (1)
                2                    (b) Superordinate:ALTAFLD
  4.2.2.1 NAME                       (c) Subordinate:None               SET 4.2.2.1
                Calculate            (d) Predecessor:INIT                EQUAL: (a)                 (2)
  4.2.2.2 FUNCTION                   (e) Successor:SELECT
                In this UOP the                                         REINTERROGATE IEN 5.3.1.3(3)
                distance which       (f) Data:
                the requesting           (f-1) Aircraft                 REPLACE IEN 3.1.2
                aircraft is able         (f-2) Airfield                 WITH 4.2.2.4               (4)
                to fly with its          (f-3) I
                remaining fuel is        (f-4) J
                calculated.              (f-5) Weather
  4.2.2.3 INPUTS
    4.2.2.3.1  INPUT NUMBER
                 1
                 Aircraft type
                 file
    4.2.2.3.2 INPUT NUMBER
                 2
                 Airfield file
  4.2.2.4 OUTPUTS
    4.2.2.4.1 OUTPUT NUMBER
                 1
                 Distance
```

(1) Moves text from report text section numbers 4.2.2 to 4.2.2.3.2 of Report No. 75 to appropriate positions in the internal structure (designated by IEN's).
(2) Provides "See Also" reference between "Calculate" and "CALC".
(3) Flags IEN 5.3.1.3 for reinterrogation for another input.
(4) Moves report text under Section 4.2.2.4 to structure under IEN 3.1.2.

Figure 3. Use of Reconciliation Commands

A reviewer (perhaps a manager) can review a document and, optionally, enter a tag for any IEN. Different tag values could indicate a suggestion or a requirement that the documentation of that element be rewritten.

Flow charting can be tied to the narrative documentation. There are two types of flow charting programs in general use. One, typified by IBM's FLOWCHART [1], requires the programmer to specify the shape of a box, its placement on the page, its connections to other boxes and its verbal context. A second, typified by the Applied Data Research Company's AUTOFLOW System [3], itself examines program text, decides on the box shape, placement and connections and takes its text from programmers' comment cards carefully embedded at the appropriate points of a program. Either system could be tied to ESDP. The text could be taken from ESDP's data base and the topology either elicited through CAINT or computed by analysis of ESDP files rather than the source program.

Cross reference files, or cross indexes, can be compiled for any aggregation of program or data documents. This will enable a considerable amount of off-line searching to be done on hard copy documents.

6. <u>User Control of the System</u>.

a. Sequence and Function Control

The responder or documentor using ESDP can exercise control over the interrogation programs in two ways: by responding to interrogation questions that ask him what he wants to do next and by non-responsive answers through which he can override the interrogation program.

It is our belief that the sequence of UOP's or UOD's documented by a programmer should be under his control and not completely pre-determined. The sequence in which a person wishes to discuss a subject must be one that is meaningful to him. Examples of determinant schemes might be: THEN path first after an IF; going through, rather than by-passing, a DO; or completely documenting one UOP before allowing a jump to another. But these rules are not necessarily correlated with meaningful execution sequences or continuity of program description.

The careful <u>interrogation program author</u> will recognize this and offer the documentor a choice of next topic whenever possible. We have done this in our experimental work by breaking the interrogation on a UOP into topics and asking, at the end of each topic, what the responder wants to consider next. He is also given control over the UOP he writes about through the //CHANGE reply described below.

Non-responsive replies (see Volume 3, Section I) are commands that the responder can issue, instead of the requested information, whenever his keyboard is open. The principal commands, and their meanings, are:

(1) //CHANGE: Responder wants to start on a different UOP which he will now be asked to name.

(2) //QUERY: Responder wants to ask a question, probably to retrieve information necessary to interpret or answer the last question. He can then enter a search on a key word or IEN basis. After a //CHANGE, he might use a //QUERY to recover the name of the UOP he wants to shift to. After the query, interrogation resumes.

(3) //REPORT: Responder wants to generate a printed report. He has the option of taking a standard format or specifying content and format. After the report is generated,the interrogation resumes.

(4) //BACK: Responder recognizes he has given the wrong answer to a previous question, wishes to go back and resume the interrogation at that point.

(5) //NO: Responder does not want to answer the question, probably because it does not apply or because he does not now have the answer. This is under CEL author control and is not allowed in all circumstances.

(6) //TO: When followed by a label, enables the responder to branch to any specified question in the interrogation. This must be used with caution because not all questions are valid entry points into the CEL program.

(7) //SIGN OFF: The responder wishes to discontinue the interrogation and resume it at another time.

With these commands, the responder can tailor the interrogation to his own views or preferences and can concentrate on one chain of reasoning at a time. The sequence of documentation does not affect the sequence of presentation in a report.

b. Unanticipated Problems.

The process of documenting a large, changing system is complex and subject to human error both in design and use. It is inevitable that situations will be encountered in use that were not foreseen in design and for which no explicit rule exists. A common form of this problem will probably arise when a programmer does not understand a computer-generated statement or when the interrogation logic or ESDP data base does not seem to fit his program.

There can be the standard conversational techniques for use when the terminal operator does not understand. A response equivalent to HELP would trigger an explanation of the last machine utterance.

When object system terms are used which are unclear, the programmer has the ESDP retrieval system which he can invoke any time his keyboard is unlocked. In this way he can quickly retrieve explanations of other object system elements that might be involved in documentation of his own element. This may be particularly important where an interrogation program author brings to a documenter's attention information on related programs or data.

Among the most difficult problems will be those concerned with how to enter information not requested by the interrogation, or in a form different from what is requested. Primarily, in this situation, we must rely on the user's knowledge and understanding of ESDP. ESDP is a complex tool and it cannot be effectively used without skill and experience. It is essential that a programmer, seeing a question printed out by the computer, understand where and how his answer is going to be used, at least in the scheduled reports - in what context the answer will appear. Knowing this, the programmer is in position to give intelligent, non-responsive answers to force the system to take the information he wants to give instead of what it apparently wants. It is not possible to do this without extensive experience. Hence, all project members should receive training and practice in the use of ESDP.

One purely mechanical aid to the befuddled programmer can be the use of a COMMENT response. It would be straightforward to allow this response to any substantive information eliciting interrogation question. It would signal that the responder wishes to add a natural language comment to the answer he has been asked for. The text to be provided would acquire the IEN of the item originally requested, with another digit or the letter C appended.

Finally, there are going to be errors in interrogation programs. The documentor should not attempt to correct them himself. Being, presumably, a trained programmer, he should be able to give a clear and descriptive report of the error to a staff member charged with interrogative programming. We must repeat that changing an interrogation program can cause great change in the ESDP data base and must be done with great care.

IV

BIBLIOGRAPHY

[1] ________, System/360 FLOWCHART (360A-SE-22X) User's Manual, Form H20-0293, IBM Corp., White Plains, N.Y. 1966

[2] ________, 1440/1460 Administrative Terminal System, Application Description, Form H20-0139, IBM Corp., White Plains, N.Y., undated

[3] ________, IBM 7090/7094 AUTOFLOW System, User's and Operator's Manual, Applied Data Research, Inc., Washington, D.C., 1967, under NASA Contract No. NAS5-10021

APPENDIX A

SAMPLE PROGRAM REPORT OUTLINE

The outline below is abbreviated far beyond the requirements for effective documentation. It was used in connection with an experimental interrogation program designed to test programming techniques. Because it is short, however, it is useful as an illustration of the process of interrogation and report generation.

Information Element No. (IEN)	Information Element Name
1.	UOP name
2.	UOP identification
2.1	Date of first interrogation
2.2	Programmer's name
2.3	Level of UOP
2.4	Modification number
2.5	Modification history
2.5.N	Modification entry
2.5.N.1	Author of modification
2.5.N.2	Date of modification
3.	Purpose of UOP
4.	Organization
4.1	Summary of organization
4.2	Superordinate UOP
4.3	Number of subordinates
4.4	List of subordinates
4.4.N	Name of subordinate

APPENDIX B

PROGRAM LOGIC INTERROGATION

QUESTION NUMBER: 001

DO YOU WISH TO

A. ADD TO AN EXISTING DATA BASE OR

B. CREATE A NEW DATA BASE.

ANSWER A OR B.

ANSWER	BRANCH TO
A	002 1/
B	002

QUESTION NUMBER: 002

THIS IS A PROGRAM DESIGN INTERROGATION.

2/ *CURRENT DATE*

WHAT IS YOUR NAME?

ANSWER	BRANCH TO
ANY	003

QUESTION NUMBER: 003

WHAT IS THE NAME OF THE UOP

YOU WISH TO DISCUSS?

ANSWER	BRANCH TO
UOP NOT KNOWN TO SYSTEM	004
UOP KNOWN TO SYSTEM	020

QUESTION NUMBER: 004

DESCRIBE THE PURPOSE OF THIS UOP.

ANSWER	BRANCH TO
ANY	005

QUESTION NUMBER; 005

SELECT THE NUMBER THAT

BEST DESCRIBES THE LEVEL OF

UOPNAME

1. SYSTEM
2. JOB
3. LOAD MODULE
4. OBJECT MODULE
5. CALL MODULE
6. GROUP
7. SEGMENT

ANSWER	BRANCH TO
1	006
2	006
3	006
4	006
5	006
6	006
7	006
OTHER	005

QUESTION NUMBER: 006

SELECT THE LETTER OF THE TOPIC
YOU WOULD LIKE TO DISCUSS.

A. PURPOSE OF *UOPNAME*
B. PROGRAM STRUCTURE
C. PROGRAM CONTROL
D. DATA
E. ERRORS
F. TESTING

ANSWER	BRANCH TO
A	008 3/
B	008
C	008
D	008
E	008
F	008
OTHER	007

QUESTION NUMBER: 007

YOU MUST ANSWER WITH A GIVEN
LETTER.

ANSWER	BRANCH TO
A	008 3/
B	008
C	008
D	008
E	008
F	008
OTHER	007

QUESTION NUMBER: 008

DO YOU KNOW THE NAME OF A
SUPERORDINATE UOP CONTAINING
UOPNAME? IF SO, GIVE ITS NAME,
OTHERWISE ANSWER NO.

ANSWER	BRANCH TO
NO	011
OTHER	009

QUESTION NUMBER: 009

WOULD YOU LIKE TO CONTINUE WITH
A. *UOPNAME* OR DISCUSS
B. *SUPER NAME*?

ANSWER	BRANCH TO
A	011
B	004
OTHER	010

QUESTION NUMBER: 010

YOU MUST ANSWER WITH A GIVEN
LETTER.

ANSWER	BRANCH TO
A	011
B	004
OTHER	010

```
QUESTION NUMBER:  011
     ARE THERE ANY SUBORDINATE UOP
     THAT ARE CONTAINED IN *UOPNAME*
                                                        ANSWER   BRANCH TO
                                                          YES       013
                                                          NO        016
                                                          OTHER     012

QUESTION NUMBER:  012
     YOU MUST ANSWER YES OR NO
                                                        ANSWER   BRANCH TO
                                                          YES       013
                                                          NO        016
                                                          OTHER     012

QUESTION NUMBER:  013
     LIST ALL UOP THAT ARE IMMEDIATELY
     CONTAINED IN *UOPNAME*
     -LIST-                                             ANSWER   BRANCH TO
                                                          //END     014
                                                          OTHER     013

QUESTION NUMBER:  014
     WOULD YOU LIKE TO TALK ABOUT
     1. *SUB UOP LIST* OR CONTINUE WITH
4/   n  *UOPNAME*?
                                                        ANSWER   BRANCH TO
                                                         UOP NAME   016
                                                         SUB UOP    016
                                                         OTHER      015

QUESTION NUMBER:  015
     YOU MUST ANSWER WITH ONE OF THE
     NUMBERS GIVEN IN THE OPTION LIST
                                                        ANSWER   BRANCH TO
                                                         UOP NAME   016
                                                         SUB UOP    016
                                                         OTHER      015

QUESTION NUMBER:  016
     NOW SUMMARIZE THE STATIC ORGANIZATION
     OF *UOPNAME* SHOWING THE RELATIONSHIP
     OF IT TO *SUPER UOPNAME* PLUS THE
     RELATIONSHIP OF *SUB UOP LIST* TO
     *UOPNAME*
                                                        ANSWER   BRANCH TO
                                                          ANY       017
```

```
QUESTION NUMBER:  017
     SELECT THE NEXT TOPIC YOU WISH
     TO COVER:
                                                       ANSWER   BRANCH TO
     A. PROGRAM CONTROL                                  A        019  3/
     B. DATA                                             B        019
     C. ERRORS                                           C        019
     D. TESTING                                          D        019
                                                         OTHER    018

QUESTION NUMBER:  018
     YOU MUST ANSWER WITH A GIVEN
     LETTER.
                                                       ANSWER   BRANCH TO
                                                         A        019  3/
                                                         B        019
                                                         C        019
                                                         D        019
                                                         OTHER    018

QUESTION NUMBER:  019
     END OF INTERROGATION.  IF YOU WISH
     TO SIGN OFF, ENTER //SIGN OFF
                                                       ANSWER   BRANCH TO
                                                        //SIGN OFF  END 5/
                                                        OTHER       019

QUESTION NUMBER:  020
     WOULD YOU STILL DESCRIBE *UOPNAME*
6/   AS BEING OF A *PROGRAM LEVEL*
     LEVEL?  PLEASE ANSWER YES OR GIVE
     NUMBER OF NEW LEVEL:
                                                       ANSWER   BRANCH TO
     1. SYSTEM                                           1        021
     2. JOB                                              2        021
     3. LOAD MODULE                                      3        021
     4. OBJECT MODULE                                    4        021
     5. CALL MODULE                                      5        021
     6. GROUP                                            6        021
     7. SEGMENT                                          7        021
                                                         YES      021
                                                         OTHER    020

QUESTION NUMBER:  021
     GIVE: CODE, START/END
                                                       ANSWER   BRANCH TO
                                                        CODE=A    021  7/
                                                        CODE=C    021
                                                        CODE=D    021
                                                        OTHER     022
```

```
QUESTION NUMBER:  022
     YOU MUST GIVE AN A OR D OR C
     FOR CODE.
                                                       ANSWER   BRANCH TO
                                                        CODE=A    021
                                                        CODE=C    021
                                                        CODE=D    021
                                                        OTHER     022
```

Footnotes

1/ At any question, in addition to the answers shown, the programmer can reply QUERY, REPORT, GO TO n or SIGN OFF at any time.

2/ The usage *CURRENT DATE* indicates that the value of the variable named could be printed here, in this case, the value of variable CURRENT DATE.

3/ This small interrogation program was written to test techniques. In several cases, information not affecting branching within the program is not elicited.

4/ When "SUB UOP LIST," the list of subordinate UOP's is printed, a line number is assigned to each number. Line number N + 1 contains "UOPNAME."

5/ SIGN OFF triggers certain housekeeping routines. The student actually has the option here, of calling for a REPORT, QUERY or GO TO a specified label.

6/ This is the beginning of an updating routine not fully implemented in this program.

7/ The programmer, here, is being asked to demand the regions on the data base he wishes to change. Coding for making the changes is not shown.

APPENDIX C

SEQUENCE TABLE FOR UPDATING INTERROGATIONS

The table below represents a portion of the table needed to implement a full updating capability with the interrogation program summarized in Appendix B. Under the heading Question Number is shown either a single question number or one or more sets of contiguous questions which must be executed in order to make modifications to the X information element specified in the first two columns.

Information Element Number (IEN)	Information Element Name	Question Number	Footnote
1.	UOP name	3	1/
2.	UOP identification		2/
2.1	Date of first interrogation	(operating system)	
2.2	Programmer's name	2	
2.3	Level of UOP	5	
2.4	Modification number	internal	3/
2.5	Modification history		2/
2.5.N	Modification entry	internal	
2.5.N.1	Author of modification	2	
2.5.N.2	Date of modification	(operating system)	
3.	Purpose of UOP	4	
4.	Organization	1-1:8-16	4/
4.1	Summary of organization	16	
4.2	Superordinate UOP	8	

4.3	Number of subordinates	internal	5/
4.4	List of subordinates	13	
4.4.N	Name of subordinate	_____	6/

FOOTNOTES

1/ All information under IEN's 1 and 2 is gathered in an initial routine which elicits the user's name and the name of the UOP. In addition at this point we obtain the current date from the operating system. The UOP name is checked to see if it is present in a list of previously documented UOP's.

If the name is in the list, the modification number (IEN 2.4) is incremented by one and the user's name and the date are stored under IEN 2.5.N.1 and 2.5.N.2, respectively. The remaining interrogation is assumed to be in the update mode.

If the UOP name does not appear in the list of known UOP's, the user's name and the date are stored under 2.2 and 2.3, respectively, and the remainder of the interrogation is carried on in the first-time mode.

2/ Because the modification history is maintained automatically, we make no provision for manual updating of this information. Hence, IEN's 2.5 and 2 cannot be revised in their entirety.

3/ This information is derived from previous modification history and the responder's indication that he wishes to update. It is not explicitly elicted by interrogation.

4/ This notation means to begin executing with the first question named (number 1, here) and continue until the second one has been executed (also number 1, hence only one question is included in this set). Then, a second set, beginning with 8 and ending with 16, is executed.

5/ This information is maintained by the interrogation program, not explicitly elicited in the interrogation.

APPENDIX D

SAMPLE INTERROGATION

Below is shown an excerpt from the actual interrogation used to produce the report shown in Appendix E. The interrogation was carried out on an IBM 2260 CRT display terminal. The text of the first few questions has been retyped for legible reproduction here.

```
THIS IS A PROGRAM DESIGN INTERROGATION  1/1/68

WHAT IS YOUR NAME?
     JOHN DOE

WHAT IS THE NAME OF THE UOP YOU WISH TO DISCUSS?
     ALTAIRFLD

DESCRIBE THE PURPOSE OF THIS UOP
     THIS IS AN ILLUSTRATIVE PROGRAM CONCEPT TO DEMONSTRATE ESDP
     TECHNIQUES.  IT IS BASED ON A FUNCTION THAT MIGHT BE PERFORMED
     BY AN ACTUAL AIR TRAFFIC CONTROL SYSTEM.  IT USES PARAMETERS
     FOR AN INDIVIDUAL AIRCRAFT, TOGETHER WITH A STORED FILE OF
     AIRCRAFT CHARACTERISTICS, TO COMPUTE THE DISTANCE THE PLANE
     CAN FLY WITH ITS REMAINING FUEL.  THEN IT SELECTS THE FIRST
     ALTERNATE AIRFIELD THAT CAN BE REACHED.

SELECT THE NUMBER THAT BEST DESCRIBES THE LEVEL OF ALTAIRFLD:
1. SYSTEM, 2. JOB, 3. LOAD MODULE, 4. OBJECT MODULE, 5. CALL MODULE,
6. GROUP, 7. SEGMENT
   5

SELECT THE LETTER OF THE TOPIC YOU WOULD LIKE TO DISCUSS
A. PURPOSE OF ALTAIRFLD, B. PROGRAM STRUCTURE, C. PROGRAM CONTROL,
D. DATA, E. ERRORS, F. TESTING
   B

DO YOU KNOW THE NAME OF A SUPERORDINATE UOP CONTAINING ALTAIRFLD
IF SO, GIVE THE NAME, OTHERWISE ANSWER NO
   TRAFFIC
```

```
WOULD YOU LIKE TO A. CONTINUE WITH ALTAIRFLD OR B. DISCUSS TRAFFIC
   A

ARE THERE ANY SUBORDINATE UOP CONTAINED IN ALTAIRFLD?
   YES

LIST ALL THE UOP THAT ARE IMMEDIATELY CONTAINED IN ALTAIRFLD -LIST-
   INITIAL
   DISTCALC
   SELECT
   OUTPUT
   //END

WOULD YOU LIKE TO TALK ABOUT:
   1. INITIAL
   2. DISTCALC
   3. SELECT
   4. OUTPUT
   5. OR CONTINUE WITH ALTAIRFLD
   5

NOW SUMMARIZE THE STATIC ORGANIZATION OF ALTAIRFLD, SHOWING THE
RELATIONSHIP OF IT TO TRAFFIC PLUS THE RELATIONSHIP OF:
   INITIAL
   DISTCALC
   SELECT
   OUTPUT
TO ALTAIRFLD
   ALTAIRFLD IS INVOKED BY A CALL STATEMENT IN TRAFFIC.  WITHIN
   ALTAIRFLD, INITIAL, DISTCALC, SELECT, AND OUTPUT ARE INVOKED
   SEQUENTIALLY IN THAT ORDER.
```

APPENDIX E

SAMPLE PROGRAM DOCUMENTATION

Below is a copy of the report generated as a result of the interrogation illustrated in Appendix D. This is an abbreviated report which results from using a short interrogation program that was primarily designed for experimenting with interrogation techniques.

```
        ESDP-CAINT PROGRAM DOCUMENTATION FOR ALTAIRFLD


            TITLE PAGE INFORMATION

1           UOP NAME
                 ALTAIRFLD
2.1         LEVEL
                 ALTAIRFLD IS A CALL MODULE UOP.
2.2         AUTHOR
                 JOHN DOE
2.1         DATE
                 680118

            EXPLANATION OF FUNCTION

3           PURPOSE
                 THIS IS AN ILLUSTRATIVE PROGRAM CONCEPT TO DEMON-
                 STRATE ESDP TECHNIQUES.  IT IS BASED ON A
                 FUNCTION THAT MIGHT BE PERFORMED BY AN ACTUAL
                 AIR TRAFFIC CONTROL SYSTEM.  IT USES PARAMETERS
                 FOR AN INDIVIDUAL AIRCRAFT, TOGETHER WITH A
                 STORED FILE OF AIRCRAFT CHARACTERISTICS, TO
                 COMPUTE THE DISTANCE THE PLANE CAN FLY WITH ITS
                 REMAINING FUEL.  THEN IT SELECTS THE FIRST
                 ALTERNATE AIRFIELD THAT CAN BE REACHED.

            BASIC FORM OF THIS UOP

4.1         SUMMARY
                 ALTAIRFLD IS INVOKED BY A CALL STATEMENT IN
                 TRAFFIC, WITHIN ALTAIRFLD, INITIAL, DISTCALC,
                 SELECT, AND OUTPUT ARE INVOKED SEQUENTIALLY
                 IN THAT ORDER.

4.2         SUPERORDINATE UOP
                 TRAFFIC
4.4         SUBORDINATE UOPS                           SUB-NAME
                 INITIAL                                  1
                 DISTCALC                                 2
                 SELECT                                   3
                 OUTPUT                                   4
```

LIST OF MODIFICATIONS TO THIS UOP

2.5 MODIFICATIONS

```
                          ALTAIRFLD--REPORT COMPLETED

ACTUAL     IEN 3.00
AIR        IEN 3.00
AIRCRA     IEN 3.00
AIRFIE     IEN 3.00
ALTAIR     IEN 4.01
ALTERN     IEN 3.00
BASED      IEN 3.00
CHARAC     IEN 3.00
COMPUT     IEN 3.00
CONCEP     IEN 3.00
DISTAN     IEN 3.00
DISTCA     IEN 4.04    IEN 4.01
ESDP       IEN 3.00
FILE       IEN 3.00
FLY        IEN 3.00
FUEL       IEN 3.00
FUNCTI     IEN 3.00
ILLUST     IEN 3.00
INDIVI     IEN 3.00
INITIA     IEN 4.04    IEN 4.01
INVOKE     IEN 4.01
ORDER      IEN 4.01
OUTPUT     IEN 4.04    IEN 4.01
PARAME     IEN 3.00
PERFOR     IEN 3.00
PLANE      IEN 3.00
PROGRA     IEN 3.00
REACHE     IEN 3.00
REMAIN     IEN 3.00
SELECT     IEN 3.00    IEN 4.04    IEN 4.01
SEQUEN     IEN 4.01
STORED     IEN 3.00
SYSTEM     IEN 3.00
TECHNI     IEN 3.00
TOGETH     IEN 3.00
TRAFFI     IEN 3.00    IEN 4.01
```

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)* Center for Exploratory Studies International Business Machines Corporation Rockville, Maryland 20850	2a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED 2b. GROUP N/A

3. REPORT TITLE

EVOLUTIONARY SYSTEM FOR DATA PROCESSING CONTROL AND USE OF THE SYSTEM

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*
None

5. AUTHOR(S) *(First name, middle initial, last name)*

Charles T. Meadow
Douglas W. Waugh
Gerald F. Conklin
Forrest E. Miller

6. REPORT DATE January 1968	7a. TOTAL NO. OF PAGES 45	7b. NO. OF REFS

8a. CONTRACT OR GRANT NO. F19628-67-C-0254 b. PROJECT NO.	9a. ORIGINATOR'S REPORT NUMBER(S) ESD-TR-68-143, Vol. II
c. d.	9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*

10. DISTRIBUTION STATEMENT

This document has been approved for public release and sale; its distribution is unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY Command Systems Division, Electronic Systems Division, Air Force Systems Command, USAF, L G Hanscom Field, Bedford, Mass. 01730

13. ABSTRACT

ESDP is a proposed system whose purpose is to acquire, store, retrieve, publish and disseminate all documentation, exclusive of graphics, concerned with a large computer programming activity. Documentation is deemed to consist, not only of final or formally published after-the-fact reports, but of working files, design and change notices, informal drafts, management reports--in fact, the entire recordable rationale underlying a programming system. Maximum attention has been concentrated on the means of acquiring and organizing documentation. Two major, complementary approaches are proposed. The first is called Program Analysis and is a process of extracting documentation directly from completed programs. The second is called Computer Assisted Interrogation and is a process of eliciting information directly from the programmer, through on-line communication terminals. The former provides canonical data about the program's structure. The latter provides explanatory material about all aspects of the program, and in the absence of canonical data, may provide tentative structural information as well. The conclusion of the study group is that ESDP is a feasible concept with present-day technology and that it will materially benefit using organizations in the production of programs and in guiding their evolution as requirements change. Its value will be greater for larger organizations, whose internal communications difficulties tend to cause truly gigantic inefficiencies. Its implementation as a support system for such projects would require a significant guantum of investment in order to produce these benefits and is predicated on the use of a computer system dedicated solely to the use of ESDP.